Towards Variable Service Compositions Using VxBPEL

Sun, Chang-ai; Aiello, Marco

Published in:
International Conference on Software Reuse

Towards Variable Service Compositions Using VxBPEL

Chang-ai Sun[1] and Marco Aiello[2]

[1] School of Computer and Information Technology, Beijing Jiaotong University
100044, Beijing, P.R. China
casun@bjtu.edu.cn
[2] Department of Computing Science, The University of Groningen
Nijenborgh 9, 9747 AG, Groningen, The Netherlands
aiellom@cs.rug.nl

Abstract. The Business Process Execution Language (BPEL) is a widely recognized executable language supporting the specification of process-oriented service compositions. However, the language is limited in addressing variable requirements in the description of business processes. We propose to construct variable and maintainable Web services compositions with VxBPEL, an extension to BPEL we developed to define variability in business process specification. We present the main concepts of VxBPEL and show how to achieve better adaptation and variability maintenance of service compositions, which is particularly desired in the context of dynamically changing business goals and processes.

1 Introduction

The growing availability of Web services both on intranets and on the Internet makes attractive to create Web service compositions to provide value-added functionalities [8]. Consider a travel agency service, for instance, which may be composed of flight and accommodation services, provided by third party providers. Since Web services themselves are deployed and executed in open and dynamic environments, the availability of service instances at run-time is an issue by itself. A composition should be flexible enough so that a flight service can be substituted by another one when a given service instance becomes unavailable. Furthermore, the user may have quite different requirements, for instance depending on the type of trip. When a traveler is arranging his/her personal trip to China, he/she would prefer to get the cheapest flight and accommodation. However, if it is a business trip, more expensive solutions may be viable. Obviously, implementing such a business process with a static Web service composition requires a great deal of recoding and manual work.

VxBPEL[7] is an extension of BPEL [4] to deal with adaptation in Web service compositions from the perspective of variability management. VxBPEL addresses the adaptive composition of Web services by providing constructs for explicitly managing variability at the composition language level, and treats the changes as first-class entities. This is a novelty with respect to current approaches [2-3,5-6], particularly those focusing on the implementation level.

In this paper, we propose to construct variable Web service compositions with VxBPEL and show how such compositions are adaptive and maintainable. We use the travel agency example for illustrational purposes. In particular, we focus here on the

H. Mei (Ed.): ICSR 2008, LNCS 5030, pp. 257–261, 2008.

explicit variability management in those cases involving dependent variation configurations, such as dependency between the flights and accommodation, or dependency between flights and frequent flyer programs.

2 Background

Variability is the ability of a software system or artifact to be extended, changed, customized, or configured for use in a specific context [9]. There are two important concepts in variability, namely variation points and variants. Variation points are locations in the design or implementation at which variations will occur, and variants are the alternatives that can be selected at variation points. Variability management includes the design, use, and maintenance of variability [1].

In order to introduce variability management into service compositions, VxBPEL extends BPEL with the constructs for defining and managing the variability. During the development of constructs for variants, variation points, and their associations, VxBPEL employs the COVAMOF variability framework [9] and adapts it to the context of Web services. The choice of COVAMOF is based on its prominent features, including treating variation points and dependencies as first-class citizens, tool support and its validation in industry.

3 Constructing Variable Compositions with VxBPEL

Let us consider the travel agency example again and model the variability with VxBPEL. There are usually several airlines which can provide a flight service required by the customer. This means that there may exist variation with the invocation of flight services. During the service composition design, we need to introduce the variation point at the place where the flight service is invoked. Fig. 1 depicts the modified BPEL process, where the activity <*in*voke> in the original BPEL process is replaced by the variation configuration. Without loss of generality, we consider two airlines, namely LH and CA. The choice between two variants is determined by the current configuration of the process.

VxBPEL supports complex realization dependencies during the service compositions. For example, an airline may have hotel partners offering discounts to the travelers. In this situation, the travel agency needs to specify the association between airlines and hotels, in order to provide the cheapest travel services to the customers who are concerned with the total travel cost. During the service composition, one can use ConfigurableVariationPoint for specifying the dependencies of such a complex service composition. Fig. 2 depicts the major segments for specifying the dependency realization between airlines and hotels. In the example, CA is the higher level variant and a set of hotel services are lower level variants for providing the discounted accommodation.

With VxBPEL, designers can focus on the main logic of business processes and, at the same time, specify the variable elements during service compositions. The specifications of VxBPEL clearly integrate main business logic and adaptation of process elements. Such service composition specifications are easily adapted, because the variability management will enable the selection of alternative variants at runtime. This is often the

```
<vxbpel:VariationPoint name= “selecting an airline service”>
  <vxbpel:Variants>
    <vxbpel:Variant name= “CA”>
      <vxbpel:VPBpelCode>
        <invoke inputVariable="FlightRequest" name="processingRequest "
          operation= "processRequest" outputVariable="requestResponse"
          partnerLink="AirlinesCA" portType="AirlinesCA:FlightProcessing">
          <target linkName="Airlines-to-Agent"/>
          <source linkName="Agent-to-Airlines/>
          </invoke>
      </vxbpel:VPBpelCode >
    </vxbpe:Variant>
    <vxbpel:Variant name= “LH”>
      <vxbpel:VPBpelCode>
        <invoke inputVariable="FlightRequest" name="processingRequest "
          operation= "processRequest" outputVariable="requestResponse"
          partnerLink="AirlinesLH" portType="AirlinesLH: FlightProcessing">
          <target linkName="Airlines-to-Agent"/>
          <source linkName="Agent-to-Airlines/>
          </invoke>
      </vxbpel:VPBpelCode >
    </vxbpel:Variant>
  </vxbpel:Variants>
</vxbpel:VariationPoint>
```

Fig. 1. The travel agency composition with variant configuration points

```
<vxbpel:ConfigurableVariationPoint id="1" defaultVariant="default">
<vxbpel:Name>... </vxbpel:Name>
<vxbpel:Rationale>...</vxbpel:Rationale>
<vxbpel:Variants>
    <vxbpel:Variant name="default ">
        <vxbpel:VariantInfo> Airline CA and its partner hotels includes the default, hotelA,
                              and hotelB which provide discounts.
        </vxbpel:VariantInfo>
      <vxbpel:RequiredConfiguration>
        <vxbpel:VPChoices>
            <vxbpel:VPChoice vpname="VP1" variant="default"/>
            <vxbpel:VPChoice vpname="VP2" variant="hotelA"/>
            <vxbpel:VPChoice vpname="VP3" variant="hotelB"/>
        </vxbpel:VPChoices>
      </vxbpel:RequiredConfiguration>
    </vxbpel:Variant>
    <!-- Another variant i.e. LH and its dependent services can be defined here. -->
</vxbpel:Variants>
 </vxbpel:ConfigurableVariationPoint>
```

Fig. 2. The illustration of service compositions with complex realization dependencies

case in the world of Web services where requirements change frequently and there is loose control over the components. The variation is supported both at compile-time and at run-time. The latter is achieved by implementing an extension to a BPEL engine to interpret the variability constructs. One may thus claim that service compositions with the VxBPEL achieve better adaptation than those with standard BPEL.

We argue that the variation of service compositions with VxBPEL is easier to understand since one can identify variation points and variants just by their prefixes. A variability management tool can aid the designer to comprehend the variation involved in service compositions. This is particularly useful when the variation of service

compositions is complex enough. Additionally, one can change variability management of service compositions by altering the variation configuration. For example, if the airline CA has more than one partner hotel (or needs to change its partner hotels), one just has to alter *vxbpel:VPChoices* to adapt to the new situation. In this sense, we claim that service compositions with VxBPEL have better maintainability than those with standard BPEL in terms of support of variation.

With VxBPEL, one can specify more variable and flexible service compositions, which thereby are able to address various dynamic changes within business processes. VxBPEL consists of BPEL native constructs and variability constructs. For a variable service composition instance, these two parts are seamlessly integrated in a VxBPEL file. Developers use BPEL native constructs for the normal service composition while the latter is used to specify the variable parts within the service composition. When these variability constructs with the prefix *vxbpel* are used, the namespace defining the VxBPEL elements must be included.

4 Concluding Remarks

Constructs provided by the current version of BPEL can be used to define fixed service compositions by specifying activities and interactions between activities. Although some structured activities such as the *switch*, may be used to select different execution paths, the selection is limited to the predefined enumerative choices and hence the configuration supported is static. When a service cannot satisfy a given QoS requirement or is unavailable, it needs to be replaced. When this occurs, the dependent services must be replaced correspondingly. Such replacement is not possible automatically with current standards. VxBPEL, on the other hand, is designed so that new variants can be introduced and managed at runtime. This allows for run-time reconfiguration and significant composition flexibility to be available within a VxBPEL process.

Acknowledgements

We thank all the contributors of the COVAMOF and the VxBPEL platforms, and Elie El-Khoury for comments. The research is partially supported by the Science and Technology Foundation of Beijing Jiaotong Univ. (Grant No. 2007RC099) and the EU Integrated Project SeCSE (IST Contract No. 511680).

References

[1] Bachmann, F., Bass, L.J.: Managing variability in software architectures. In: Proceedings of ACM SIGSOFT Symposium on Software Reusability, pp. 126–132 (2001)

[2] Charfi, A., Mezini, M.: AO4BPEL: An Aspect-Oriented Extension to BPEL. World Wide Web Journal: Recent Advances on Web Services (special issue) 10(3), 309–344 (2007)

[3] Colombo, M., Nitto, E.D., Mauri, M.: SCENE: a service composition execution environment supporting dynamic changes disciplined through rules. In: Proceedings of ICSOC 2006. LNCS, vol. 4292, pp. 191–202. Springer, Heidelberg (2006)

[4] Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Weerawarana, S.: Business process execution language for Web services, Version 1.1 (2003)
[5] Ezenwoye, O., Sadjadi, S.M.: TRAP/BPEL: A Framework for Dynamic Adaptation of Composite Services, http://www.cs.fiu.edu/~sadjadi/Publications/TechRep-FIU-SCIS-2006-06-02-TRAP-BPEL.pdf
[6] Erradi, A., Maheshwari, P.: AdaptiveBPEL: a Policy-Driven Middleware for Flexible Web Services Compositions. In: Proceedings of Middleware for Web Services (MWS) (2005)
[7] Koning, M., Sun, C., Sinnema, M., Avgeriou, P.: VxBPEL: Supporting variability for Web services in BPEL. In: Information and Software Technology. Elsevier, Amsterdam, http://dx.doi.org/10.1016/j.infsof.2007.12.002
[8] Papazoglou, M.P.: Web services technologies and standards. ACM Computing Surveys (submitted, 2006), http://infolab.uvt.nl/pub/papazogloump-2006-97.pdf
[9] Sinnema, M., Deelstra, S., Hoekstra, P.: The COVAMOF derivation process. In: Morisio, M. (ed.) ICSR 2006. LNCS, vol. 4039, pp. 101–114. Springer, Heidelberg (2006)